Lab3

Ahmed Alhasan, Yash Pawar, Mohsen Pirmoradiyan

2/8/2020

Question 1: Cluster sampling

In this question we are asked to generate a sampling function to select 20 cities among the Sweden cities. The selection, however, is not supposed to be uniform. The population of the cities must be taken into account as the proportionality in sampling. In other words, the proportion of the populations to the total population of the country must be amounted as the propability of being selected. So the expectd is that the higer the propabilty(population proportion), the higher the chance to be selected. To meet the goal we ordered the cities based on their population, then the cumulative sum of populations was computed. Each city can the be assigned to the interval started from the previous cumulative sum to its own cumulative sum. Hence the higher the population a city has, the wider the interval that city belongs to and so the higher chance to be selected.

1) First Method

```
pop = read.csv2("../Data/population.csv", stringsAsFactors = FALSE)
mydata = pop
mySample2 <- function(size, x){
  #A dataframe for output
  mycities = data.frame(Cities=as.character(), Population = as.numeric()
                        , stringsAsFactors = F)
  #Creating a new data to be changed.
  new_x = x
  #Changing the format in case if the format is "factor"
  new_x$Municipality = as.character(new_x$Municipality)
  new_x$Population = as.numeric(new_x$Population)
  #Adding a new column to comute cumulative sum
  new_x$cum_sum = 0

  for (i in 1:size) {
    #Sorting the data with respect to population
    new_x = new_x[order(new_x$Population),]
    #Calculating the cumulative sum
    new_x$cum_sum = cumsum(new_x$Population)

    #Randomly select a number between 1 and total population
    d = runif(1, 1, max = max(new_x$cum_sum))
    #Check to find which interval the randomly selected number belongs to
    #First we check if it ibelongs to first row with the lowest population
    if(d <= new_x$Population[1]){
      city = new_x$Municipality[1]
```

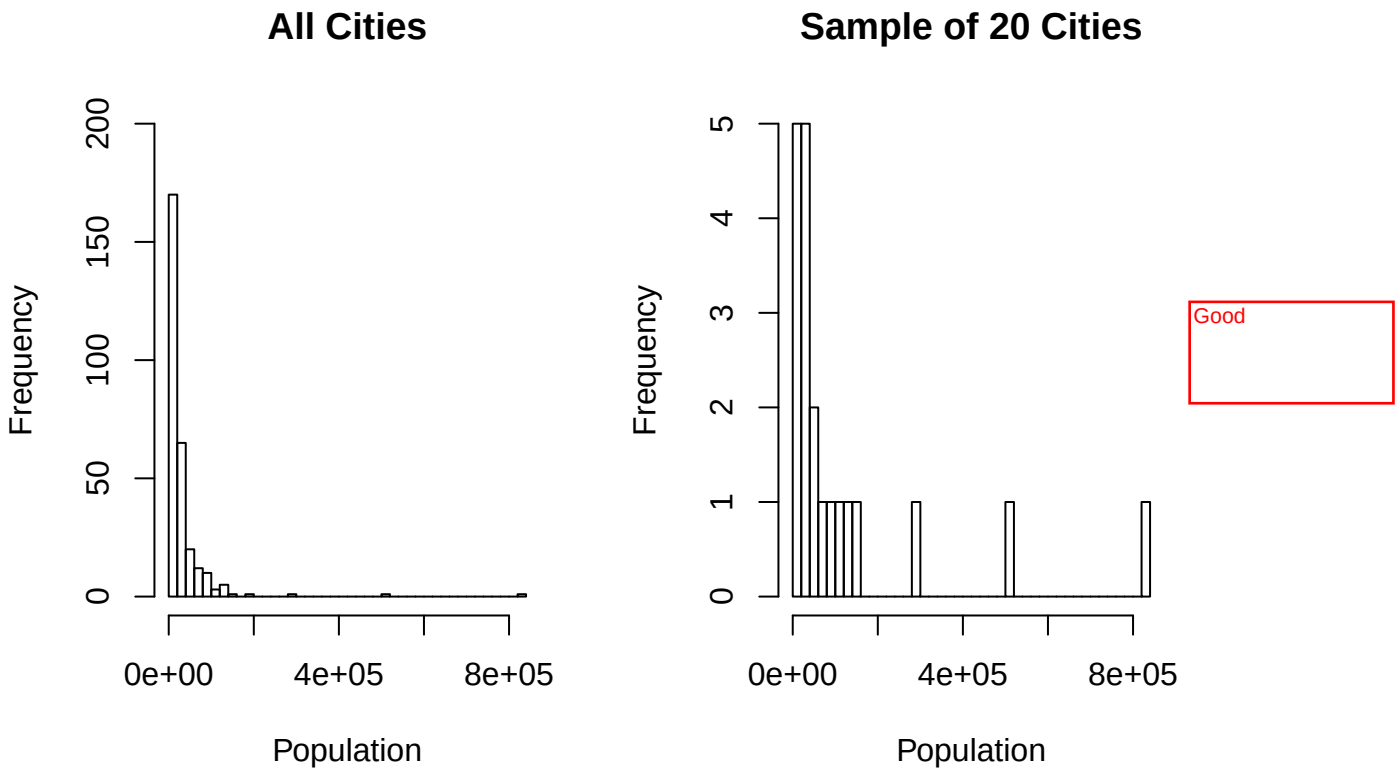
```
      population = new_x$Population[1]
      mycities = rbind(mycities, data.frame(Cities = city, Population=population,
                                            stringsAsFactors = F))
      new_x <- new_x[-1,]

    }else{
      #Then we check for the rest of the intervals
      for (j in 2:nrow(new_x)) {

        if(d <= new_x$cum_sum[j] & d > new_x$cum_sum[j-1] ){
          city = new_x$Municipality[j]
          population = new_x$Population[j]
          mycities = rbind(mycities, data.frame(Cities = city, Population=population,
                                                stringsAsFactors = F))
          new_x <- new_x[-j,]
          break
        }
      }
    }
  }

  return(mycities)
}
```

The selected cities are:

Cities	Population
Jönköping	126331
Göteborg	507330
Umeå	114075
Malmö	293909
Piteå	40860
Höör	15261
Gislaved	29212
Uddevalla	51518
Gävle	94352
Stockholm	829417
Vadstena	7420
Sunne	13345
Kristianstad	78788
Dorotea	2900
Danderyd	31150
Sundbyberg	37722
Karlshamn	30918
Strängnäs	32024
Bjuv	14813
Linköping	144690

As the above list shows the almost the cities with high population have been selected.

The histogram of all cities vs. the selected sities:



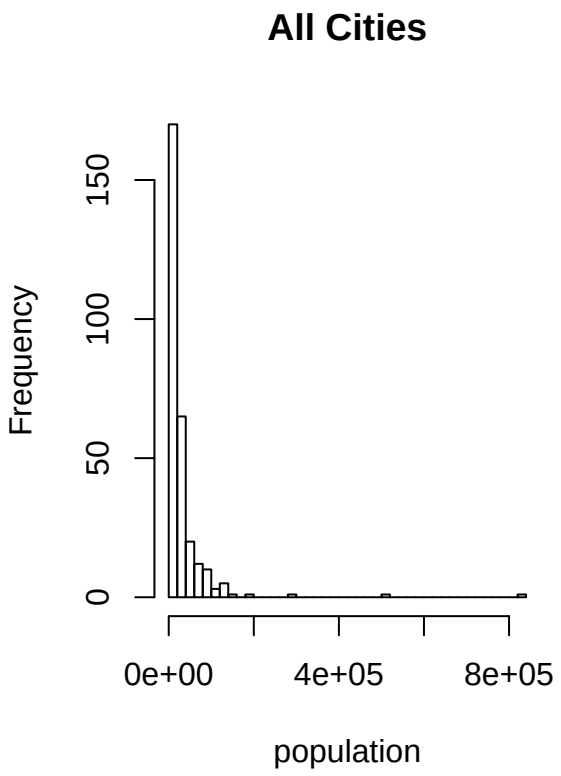
The histogram of the selected cities show that the weights of the higer populated cities are more than those of the lower size cities. Considering the objective of the question to randomly selecting the cities proportional to their population, such a rondomized selection seems to be reliable.

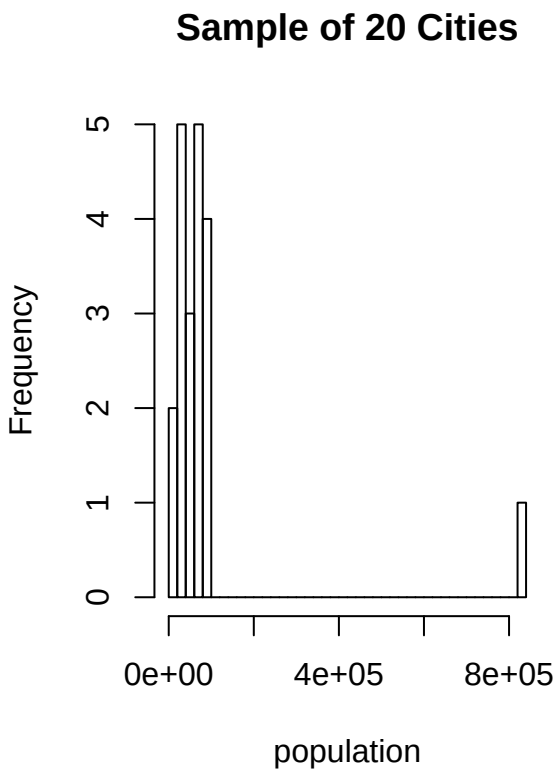
2) Second Method

```
population <- read.csv2("../Data/population.csv", stringsAsFactors = FALSE)
sampler <- function(data, n){
  #taking the cumsum for the probabilities means
  #we put them in a period from 0 to 1 without overlapping
  data$prob <- cumsum(data[,2] / sum(data[,2]))

  cities <- as.character()
  pop    <- as.numeric()
  for(i in 1:n){
    rand <- runif(1)
    #applying the generalized inverse distribution function which takes only the nearest
    #CMD value that is equal or larger than the probability value ..cities with higher population
    #cover longer periods so they have higher chances to be closest to the roll
    cities[i] <- data[which.min((data$prob-rand)[which((data$prob-rand)>=0)]),][[1]]
    pop[i]    <- data[which(data[,1] == cities[i]),][[2]]
    data      <- data[-which(data[,2] == pop[i]),]
  }
```

```
  res <- data.frame(Municipality = as.character(cities), Population = as.numeric(pop))
  return(res)
}
```

```
##    Municipality Population
## 1      Botkyrka      81195
## 2      Danderyd      31150
## 3         Ekerö      25095
## 4       Haninge      76237
## 5      Huddinge      95798
## 6      Järfälla      65295
## 7       Lidingö      43445
## 8         Nacka      88085
## 9     Norrtälje      55927
## 10      Nykvarn       9227
## 11    Nynäshamn      25781
## 12        Salem      15313
## 13      Sigtuna      39219
## 14   Sollentuna      63347
## 15        Solna      66909
## 16    Stockholm     829417
## 17   Sundbyberg      37722
## 18   Södertälje      85270
## 19       Tyresö      42602
## 20         Täby      63014
```





- Since we are trying to sample from a discrete distribution we obtain the CDF by cumulatively adding up the individual probabilities for the cities and selecting the nearest city that has it's CMD value equal or larger than the probability generated from the uniform distribution of the CDF, based on the generalized inverse distribution function:

$$F^{-1}(U) = inf\{x \in \mathbb{R} : F(x) \geq U\}$$

Actually in your method 1 you are also implicitly computing this function.

Question 2: Different distributions

1)inverse CDF method

The double exponential (Laplace) distribution:

$$DE(\mu, \alpha) = \frac{\alpha}{2} \exp(-\alpha|x - \mu|)$$

So for $DE(0, 1)$, the probability density function is:

$$f(x) = \frac{1}{2} \exp(-|x|)$$

when $x \geq 0$: $f(x) = \frac{1}{2} \exp(-x)$

when $x < 0$: $f(x) = \frac{1}{2} exp(x)$

By integrating $f(x)$ over the related domain the CDF can be obtained:

For $x < 0$:

$F(x) = \frac{1}{2} \int_{-\infty}^{x} \exp(s)ds = \frac{1}{2} \exp(x)$

For $x \geq 0$:

$F(x) = \frac{1}{2} \int_{-\infty}^{0} \exp(s)ds + \frac{1}{2} \int_{0}^{x} \exp(-s)ds = \frac{1}{2} - \frac{1}{2}[\exp(-x) - 1] = 1 - \frac{1}{2} \exp(-x)$

To obtain the inverse CDF we solve these equations for x:

For $x \geq 0$:

$$U = 1 - \frac{1}{2} \exp(-x) \Rightarrow 2(1 - U) = \exp(-x) \Rightarrow x = -\ln 2(1 - U)$$

$$x \geq 0 \Rightarrow \ln 2(1 - U) \leq 0 \Rightarrow 0 < 2(1 - U) \leq 1 \Rightarrow 0 < 1 - U \leq \frac{1}{2} \Rightarrow -1 < -U \leq -\frac{1}{2} \Rightarrow \frac{1}{2} \leq U < 1$$

And for $x < 0$:

$$2U = \exp(x) \Rightarrow x = \ln(2U)$$

$$x < 0 \Rightarrow 0 < 2U < 1 \Rightarrow 0 < U < \frac{1}{2}$$

Therefore, the inverse CDF is as follows:

for $0 < U < \frac{1}{2}$

$$F^{-1}(U) = \ln(2U)$$

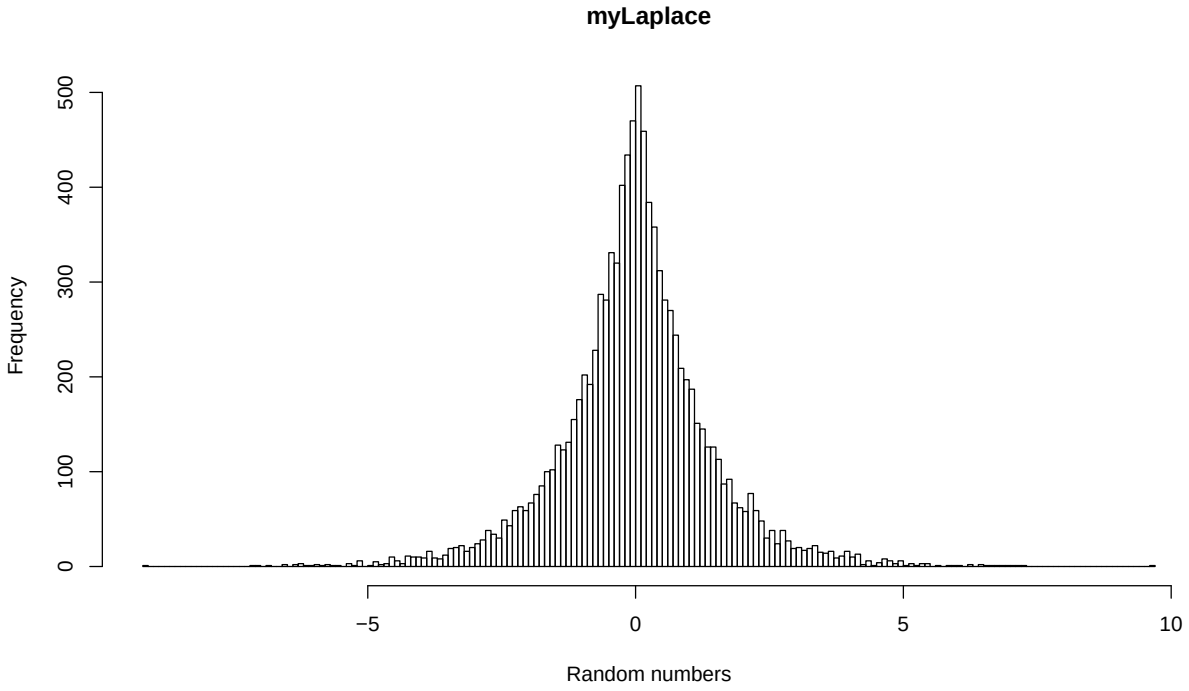
Good!

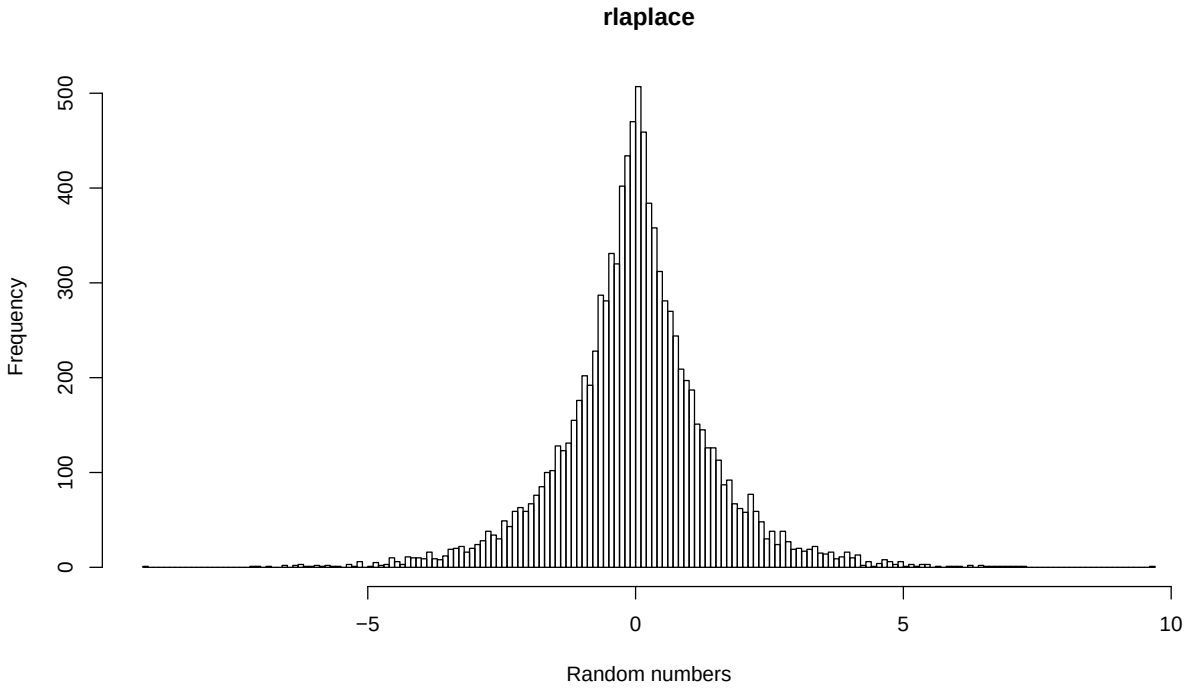
for $\frac{1}{2} \leq U < 1$

$$F^{-1}(U) = -\ln 2(1 - U)$$

Now we can write a function to generate random numbers from double exponentioan distribution.

The following histograms show two distributions of generated numbers implementing our function and standard R function(*rlaplace*() from *rmtil* package). Our function seems to has simulated the double exponential distribution properly.


myLaplace
Frequency
0
100
200
300
400
500
−5
0
5
10
Random numbers


rlaplace
Frequency
0
100
200
300
400
500
−5
0
5
10
Random numbers

2)Acceptance/rejection method

The target density function is the standard normal distribution($Z \sim N(0,1)$):

$$f(x) = \frac{1}{\sqrt{2\pi}} \exp(-x^2/2)$$

The majorizing density function is the laplace density($DE(0,1)$):

$$g(x) = \frac{1}{2} \exp(-|x|)$$

we define the function $h(x)$ as the ratio $f(x)/g(x)$ as follows:

$$h(x) = f(x)/g(x) = \frac{2}{\sqrt{2\pi}} \exp(|x| - x^2/2)$$

to find the majorizing constant, c, we will find the maximum of $h(x)$, since c is the upper bound for this function:

$$h'(x) = \frac{2}{\sqrt{2\pi}}(-x + \frac{x}{|x|}) \exp(|x| - x^2/2) = 0$$

$$\Rightarrow -x + \frac{x}{|x|} = 0$$

$\Rightarrow$ for $x \geq 0$ $x = 1$

and

for $x < 0$ $x = -1$

Therefore the maxumim value of $h(x)$ will be:

$$max(h(x)) = \frac{2}{\sqrt{2\pi}} \exp(1 - 0.5) = \frac{2\sqrt{e}}{\sqrt{2\pi}}$$

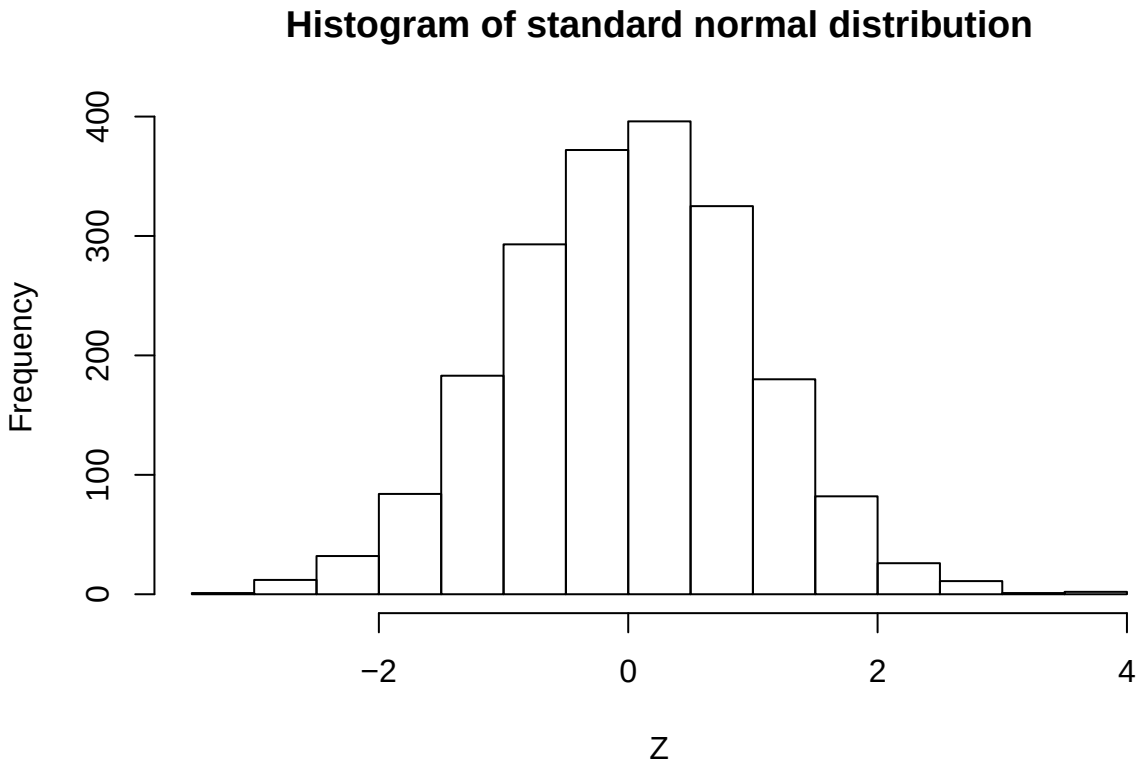
Hence $c =$

```
## The majorizing constant: 1.315489
```

Correct

Haveing computed the constant c, we have the ratio $\frac{f(x)}{cg(x)}$ as the condtion term, hence we can generate random number from the target distribution implementing accept-reject method. In previous task we wrote a function to generate random number from the laplace distribution implementing inverse transform method. In this task we will be using that function since the laplace function is now a majorizing density function. Firstly, we should generate random nember from this density, then implementing the conditioning term, either accept it as the desired number for our target density or reject it depending on holding the condition or not. The condition will be passed if the uniformly generated number(U) is less than or equal to $\frac{f(y)}{cg(y)}$

```
## The number of rejections:  629
```



The number of times required to the event "Acceptance" occured is a random variable which has a geometric distribution with success probabbility of:

$$p = Pr(U \leq \frac{f(Y)}{cg(Y)} | Y = y)$$

Considerind the density distribution $g(Y)$ and the value $\frac{f(y)}{cg(y)}$ we have:

$$p = \int_{-\infty}^{\infty} \frac{f(y)}{cg(y)} g(y) dy = \frac{1}{c} \int_{-\infty}^{\infty} f(y) dy = \frac{1}{c}$$

Hence the expected value of "Acceptance" will be $1/c$ which is:

```
## 0.7601735
```

Which is regarded as acceptance rate. Therefore the "Rejection rate" will be $1 - 1/c$:

Correct

```
## 0.2398265
```

In our implementation of the algorithm the number of times which the event "Rejection" occured was obtained as:

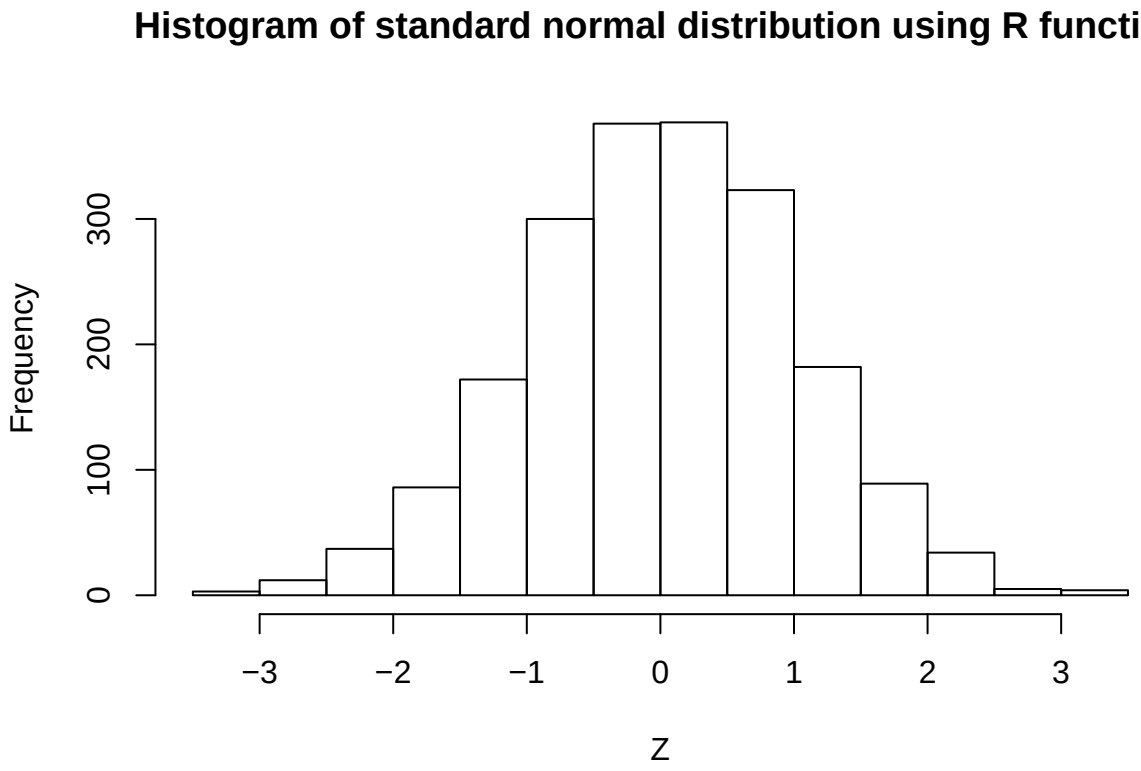
```
## [1] 629
```

considering the 2000 generated random numbers, the rejection rate was629/2629 which is:

```
## The rejection rate:  0.2392545
```

This value is very close to the Expected value of Rejection

Generating random numbers from $N(0,1)$ implementing the R function($rnorm()$):



Comparing this histogram with the one we obtained using the acceptance-rejection methods, we can conclude that the two results look similar, hence our function is reliable.

Appendix

```
library(rmutil)
knitr::opts_chunk$set(echo = FALSE)
pop = read.csv2("../Data/population.csv", stringsAsFactors = FALSE)
mydata = pop
mySample2 <- function(size, x){
  #A dataframe for output
  mycities = data.frame(Cities=as.character(), Population = as.numeric()
                        , stringsAsFactors = F)
  #Creating a new data to be changed.
  new_x = x
  #Changing the format in case if the format is "factor"
  new_x$Municipality = as.character(new_x$Municipality)
  new_x$Population = as.numeric(new_x$Population)
  #Adding a new column to comute cumulative sum
  new_x$cum_sum = 0

  for (i in 1:size) {
    #Sorting the data with respect to population
    new_x = new_x[order(new_x$Population),]
    #Calculating the cumulative sum
    new_x$cum_sum = cumsum(new_x$Population)

    #Randomly select a number between 1 and total population
    d = runif(1, 1, max = max(new_x$cum_sum))
    #Check to find which interval the randomly selected number belongs to
    #First we check if it ibelongs to first row with the lowest population
    if(d <= new_x$Population[1]){
      city = new_x$Municipality[1]
      population = new_x$Population[1]
      mycities = rbind(mycities, data.frame(Cities = city, Population=population,
                                            stringsAsFactors = F))
      new_x <- new_x[-1,]

    }else{
      #Then we check for the rest of the intervals
      for (j in 2:nrow(new_x)) {

        if(d <= new_x$cum_sum[j] & d > new_x$cum_sum[j-1] ){
          city = new_x$Municipality[j]
          population = new_x$Population[j]
          mycities = rbind(mycities, data.frame(Cities = city, Population=population,
                                                stringsAsFactors = F))
          new_x <- new_x[-j,]
          break
        }
      }
    }
  }

  return(mycities)
}
```

```r
set.seed(12345)
selected = mySample2(size = 20, x = mydata)
knitr::kable(selected)
par(mfrow=(c(1,2)))
hist(mydata[[2]],breaks = 40, xlab = "Population", ylim = c(0,200), main = "All Cities")
hist(selected[[2]],breaks = 40, xlab = "Population", main = "Sample of 20 Cities")


population <- read.csv2("../Data/population.csv", stringsAsFactors = FALSE)
sampler <- function(data, n){
  #taking the cumsum for the probabilities means
  #we put them in a period from 0 to 1 without overlapping
  data$prob <- cumsum(data[,2] / sum(data[,2]))

  cities <- as.character()
  pop    <- as.numeric()
  for(i in 1:n){
    rand <- runif(1)
    #applying the generalized inverse distribution function which takes only the nearest
    #CMD value that is equal or larger than the probability value ..cities with higher population
    #cover longer periods so they have higher chances to be closest to the roll
    cities[i] <- data[which.min((data$prob-rand)[which((data$prob-rand)>=0)]),][[1]]
    pop[i]    <- data[which(data[,1] == cities[i]),][[2]]
    data      <- data[-which(data[,2] == pop[i]),]
  }
  res <- data.frame(Municipality = as.character(cities), Population = as.numeric(pop))
  return(res)
}


sampler(data = population, n = 20)
par(mfrow=c(1,2))
hist(population$Population, breaks = 50, xlab = "population", main = "All Cities")
hist(sampler(data = population, n = 20)$Population, breaks = 50, xlab = "population", main = "Sample of


myLaplace = function(size){
  u = runif(size,0,1)
  return(sapply(u, function(i){
    if(i < 0.5){
      return(log(2*i))
    }else{
      return(-log(2*(1-i)))}
  }
  )
  )

}

par(mfrow=c(2,1))
set.seed(12345)
```

```r
de=myLaplace(10000)
hist(de, breaks = 250, xlab = "Random numbers", main = "myLaplace")

set.seed(12345)
hist(rlaplace(10000), breaks = 250, xlab = "Random numbers", main = "rlaplace")
#Comuting the majorizing constant
c = 2*sqrt(exp(1))/sqrt(2*pi)
cat("The majorizing constant:", c)
#Acceptance-Rejection function
myAccept = function(size){

  R=0
  Y=vector(length = size)
  for (i in 1:size) {
    repeat {
      y = myLaplace(1)
      h = (1/sqrt(exp(1)))*exp(abs(y)- (y^2/2))
      U = runif(1)
      if(U <= h){Y[i]=y; break}
      else{R = R+1}
    }
  }
  return(list(Y=Y, Reject=R))
}

set.seed(12345)
Z = myAccept(2000)

cat("The number of rejections: ",Z$Reject)


hist(Z$Y, xlab = "Z", main = "Histogram of standard normal distribution")


#1/c
cat(1/c)
#1-1/c
cat(1-1/c)
# The number of times that rejections occured
Z$Reject
cat("The rejection rate: ", Z$Reject/(Z$Reject+2000))
set.seed(12345)
Z = rnorm(2000)

hist(Z, main = "Histogram of standard normal distribution using R function")
```